sensors

MDPI

Article

Distributed Multi-Robot Information Gathering under Spatio-Temporal Inter-Robot Constraints†

Alberto Viseras [1,*], Zhe Xu [2] and Luis Merino [3]

[1] German Aerospace Centre (DLR), Oberpfaffenhofen, 82234, Germany
[2] Australian Centre for Field Robotics (ACFR), Sydney, NSW 2006, Australia; z.xu@acfr.usyd.edu.au
[3] School of Engineering, Universidad Pablo de Olavide (UPO), Seville, 41013, Spain; lmercab@upo.es

† This manuscript is extension version of the conference paper: Viseras, A.; Xu, Z.; Merino, L. Distributed Multi-Robot Cooperation for Information Gathering under Communication Constraints. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May, 2018.

* Correspondence: Alberto.ViserasRuiz@dlr.de

Received: 4 November 2019; Accepted: 28 December 2019; Published: date

Abstract: Information gathering (IG) algorithms aim to intelligently select the mobile robotic sensor actions required to efficiently obtain an accurate reconstruction of a physical process, such as an occupancy map, a wind field, or a magnetic field. Recently, multiple IG algorithms that benefit from multi-robot cooperation have been proposed in the literature. Most of these algorithms employ discretization of the state and action spaces, which makes them computationally intractable for robotic systems with complex dynamics. Moreover, they cannot deal with inter-robot restrictions such as collision avoidance or communication constraints. This paper presents a novel approach for multi-robot information gathering (MR-IG) that tackles the two aforementioned restrictions: (i) discretization of robot's state space, and (ii) dealing with inter-robot constraints. Here we propose an algorithm that employs: (i) an underlying model of the physical process of interest, (ii) sampling-based planners to plan paths in a continuous domain, and (iii) a distributed decision-making algorithm to enable multi-robot coordination. In particular, we use the max-sum algorithm for distributed decision-making by defining an information-theoretic utility function. This function maximizes IG, while fulfilling inter-robot communication and collision avoidance constraints. We validate our proposed approach in simulations, and in a field experiment where three quadcopters explore a simulated wind field. Results demonstrate the effectiveness and scalability with respect to the number of robots of our approach.

Keywords: robotics; distributed multi-agent systems; information gathering; Gaussian processes

1. Introduction

Information gathering (IG) is a key task in many robotic applications such as, e.g., magnetic field mapping [1], environmental monitoring [2], or wind field mapping [3]. The IG task can clearly benefit from distributed multi-robot coordination strategies: First, by means of parallelization, as tasks could be split between robots, and second, in terms of robustness, as tasks of a faulty robot could be overtaken by the rest of the team.

A common approach often used in the literature to solve multi-robot information gathering (MR-IG) tasks is to use an underlying model of the physical process under study that, together with an information-theoretic metric, is employed to predict the impact of certain robot actions and states (see, e.g., [1,4]). In particular, in this work we assume Gaussian processes (GPs) for regression [5] as underlying model, and mutual information (MI) [6] as information metric.

GPs are state-of-the-art models to represent spatio-temporal fields [1–3,5]. Furthermore, the use of GPs together with MI has been shown to significantly outperform former MR-IG algorithms [7,8]. However, most state-of-the-art algorithms employ discretization of the robots state and action spaces, and do not take into account robots' dynamics. This makes such strategies computationally intractable for robotic systems with complex dynamics, like, e.g., an aircraft. This limits the applicability of state-of-the-art algorithms to a reduced class of "simple" robots. In addition, most MR-IG algorithms do not take into account inter-robot constraints like, e.g., inter-robot collision avoidance or communication constraints. Such limitations preclude state-of-the-art algorithms to be transferred from simulations to real world experiments.

In this paper we propose an algorithm that tackles the two aforementioned issues: generality of the strategies, and handling of inter-robot constraints. Our algorithm consists of two major blocks. First, we use rapidly-exploring random trees (RRT) to design an algorithm that can be generalized to multiple classes of robots, as RRT is a state-of-the-art strategy to plan paths in continuous high dimensional spaces. Second, we employ distributed constraint optimization (DCOP) techniques to handle inter-robot constraints in a distributed fashion. In particular, we employ Max-sum for multi-robot coordination [9]. Max-sum, in contrast to other DCOP techniques (see [9] for an overview), makes efficient use of the computational and communication resources. In addition, it offers approximate solutions that are close to optimal for many applications of interest like, e.g., exploration, or tracking [4], as well as a solution for cooperative games [10] that can be also used to model multi-robot cooperation for IG tasks [11].

In this paper, we build on RRT and max-sum, together with GPs and MI, to derive a novel active perception strategy that allows multiple robots to autonomously gather information of a physical process of interest. Specifically, in opposition to state-of-the-art strategies (see Section 2), our proposed strategy is able (i) to account for robots with complex dynamics that operate in a continuous environment; (ii) to solve an IG task in a distributed fashion with local inter-robot communication; and (iii) to incorporate spatial and temporal inter-robot constraints.

We evaluate our approach in simulations where multiple aerial vehicles, which are subject to collision avoidance and communication constraints, cooperate to map a wind field. In addition, we carry out an outdoor field experiment where a fleet of three aerial robots explore a simulated wind field. Results of the experiment demonstrate the effectiveness, as well as the online realization of the algorithm.

Let us also point out that the work presented here is largely based on our previous publication [12], yet it extends the latter in several important respects. In particular,

- We introduce a discussion on state-of-the-art information metrics for IG. This discussion allows us to motivate and proof, empirically through simulations, our specific choice.
- We introduce algorithmic approximations that permit an online realization of our algorithm. In addition, we present a detailed analysis of our algorithm's computational complexity.
- We include an inter-robot collision avoidance constraint that permits collision-free IG.
- We extend the simulations setup from four to up to eight robots. This allows us to show the algorithm's scalability as the number of robots increases.
- We analyze in detail the effect of algorithm parameters in the algorithm's performance.
- We provide a full description of the experimental setup, and additional experimental results.

The remainder of the paper is organized as follows. First, we review the related work in Section 2. Then we state the problem formally in Section 3. Next we introduce in Section 4 the methods in which our algorithm builds. Our algorithm relies on an information metric to decide robots' action. Therefore, we include in Section 5 a discussion about the suitability of several information metrics for our specific problem. This is followed in Sections 6 and 7 by a detailed explanation of the different subsystems that compose our system, and an assessment of the algorithm's computational complexity, respectively. Then we test the algorithm in simulations in Section 8, and verify in Section 9 its performance with a

field experiment in which three quadcopters explore an unknown simulated wind field. We finalize with a summary and outlook of the paper in Section 10.

2. Related Work

Multi-robot IG is a topic that has attracted lots of interest in the last years, and many strategies have been proposed recently. Here we focus on model-based strategies that use GPs as underlying model of a process of interest. In [6] the authors present a multi-robot exploration algorithm that exploits the properties of GPs as underlying model and of MI as information metric. By leveraging the submodularity property of MI, the authors derive a sequential greedy algorithm that offers worst case guaranties. However, the algorithm proposed in [6] is centralized, as it requires robots to have access to information from the complete team.

A solution that goes beyond a centralized architecture was proposed in [13], where the authors propose a strategy that is semi-decentralized. That is, the algorithm combines a centralized and a decentralized processing: On the one hand, a master robot selects a set of observation points that maximize a joint entropy. On the other hand, robots perform data fusion in a decentralized fashion.

In contrast to [6,13], here we argue for the use of a decentralized architecture to decide robots movement. Multiple works proposed decentralized IG methods with GPs; see, e.g., [1,4,13,14]. The previous techniques [1,4,13,14] employ a discretization of the robot state and action spaces and search-based planning algorithms for exploration. Thus, they cannot consider kinematic constraints associated to the robot motion for non-holonomic vehicles, like fixed-wing aircrafts.

Informative path planning techniques typically encompass algorithms that aim to plan a path which is both feasible, given a robot's dynamical constraints, and optimal with respect to some information quality metric. Single-robot approaches for informative path planning in continuous spaces have been proposed in [15–20]. However there is little work in the literature that propose multi-robot informative path planning algorithms. In [17] the authors present a multi-robot sampling-based informative path planner. However, [17] is limited to the particular case of tracking applications. In [21] a multi-robot information gathering method is proposed, which is also designed for tracking applications.

Extending the informative path planning problem to multi-robot settings involves two key tasks. First, robots must be able to cooperate to maximize the fleet's information gain. Furthermore, robots must be able to handle mission-specific spatio-temporal inter-robot constraints. To tackle the two aforementioned tasks, [8] and [22] divide the problem in task and path planning. In particular, [8] and [22] consider at task level GPs and a MI utility function to determine informative points to be visited in a multi-robot exploration scenario. Then, paths are planned towards those points in a centralized way. In contrast, we propose in this paper an algorithm that allows robots to plan paths in a decentralized fashion using inter-robot local communication.

One fundamental aspect in multi-robot IG is how to handle inter-robot spatio-temporal constraints. In [23] the authors propose a method to handle such constraints for a target search problem. Specifically, they employ the augmented Lagrangian method for the multi-robot cooperation. The use of the Lagrangian methods typically requires constraints and the objective function, which guides robots movement, to be differentiable. This limits the applicability of the algorithm to a selected class of objective functions and constraints, and limits the generality of the algorithm. On the contrary, this paper presents a method that is able to incorporate a large class of objective functions; the only requirement is that the global objective function can be expressed as the sum of the individual objective functions of each of the agents. To this end, we employ the max-sum algorithm from [24], and extend it to allow exploration in a continuous space for robots with kinematic, kinodynamic and spatial-temporal constraints.

3. Problem Statement

We consider here a problem of exploring an *a priori* unknown physical process with N cooperative robots autonomously and as accurately as possible, in the sense of minimizing the Root Mean Squared Error (RMSE) between a process estimate (given by a GP model) and the (unknown) ground truth. Additionally, exploration should be efficient, in the sense of minimizing the RMSE as fast as possible, provided the (i) available resources, and (ii) complex constraints such as inter-robot collision avoidance and communication constraints.

To achieve this, we make a few simplifying assumptions. Specifically, we assume the following:

1. The physical process of interest can be modeled with a GP sufficiently well.
2. This process is time-invariant during the IG task.
3. The robot positions are known exactly and are noise-free. That is, we assume that there exists an external positioning system that provides us with a highly accurate localization, e.g., a Real-Time Kinematic navigation for global positioning systems (GPS-RTK) for outdoor scenarios, or a motion tracking system for indoor environments. Uncertainty in positioning could also be accounted for using GPs [25], but it is out of the scope of this work.

Additionally, our problem is subject to the following physical constraints:

4. The robot i, with $i = 1, ..., N$, motion model is given by a known function $\mathbf{x}_i(t + \Delta t) = \mathbf{f}(\mathbf{x}_i(t), \mathbf{u}_i)$ that relates the robot's current position $\mathbf{x}_i(t)$ and future position $\mathbf{x}_i(t + \Delta t)$ given a control input $\mathbf{u}_i$, where Δt is the duration of a single time step.
5. Robots can only directly communicate if they are neighbors, i.e. if they are separated by less than a distance r_c. This defines a robots communication graph $\mathcal{G}_c(\mathcal{V}_t, \mathcal{E}_t)$ at time t, with $\mathcal{V}_t = \{\mathbf{x}_1(t), ..., \mathbf{x}_N(t)\}$, and $\mathcal{E}_t = \{(\mathbf{x}_i(t), \mathbf{x}_j(t)) : i, j \in \{1, ..., N\}, i \neq j, ||\mathbf{x}_i(t) - \mathbf{x}_j(t)|| \leq r_c\}$. We also assume that there exists an underlying communication protocol, like, e.g., TCP/IP, which ensures an error-free data transmission.

Furthermore, robots must fulfill the following mission-related constraints:

6. *Inter-robot collision avoidance*: two robots collide if they are separated less than a distance r_s. Distance r_s shall take into account robots' shape, as well as the potential uncertainty in robot positions.
7. *Network connectivity*: the network of robots requires a periodic connectivity, with a maximum disconnection time of $k_c \Delta t$ seconds, where k_c is a constant that denotes a number of time steps.

Let us now introduce some notation that we will use in the remainder of the paper. The position of robot i will be denoted by $\mathbf{x}_i(t) \in \mathcal{X}_{free}$, where $\mathcal{X}_{free} \in \mathbb{R}^{d_s}$ corresponds to the free space in the robot's configuration space, with d_s being the dimensionality of the environment in which a process of interest takes place. The physical process at position $\mathbf{x} \in \mathcal{X}_{free}$ is denoted as $y(\mathbf{x}) \in \mathbb{R}$. Typically, however, a process is not observed directly, but is measured using some sensors. Here we assume a simple sensor model that represents a measured process as $z(\mathbf{x}) = y(\mathbf{x}) + \epsilon(\mathbf{x})$, where $z(\mathbf{x})$ is a process sample, $y(\mathbf{x})$ is the unobserved true process value, and $\epsilon(\mathbf{x})$ is a random noise. In the following we will assume that, for different measurements, noise samples $\epsilon(\mathbf{x})$ are independent and identically distributed according to $\mathcal{N}(0, \sigma_n^2)$; i.e., they follow a Gaussian distribution with zero mean and variance σ_n^2.

The afore-described problem requires an infinite number of steps to be completed, which corresponds to an infinite horizon IG task. However, for the sake of computational feasibility, a common approach in IG is to divide an infinite horizon problem into multiple finite horizon problems that can be solved individually and sequentially. In particular, here we assume an horizon of k_c time steps. That is, every k_c time steps robots solve an IG problem to find a set of paths $P = \{\mathcal{P}_1, ..., \mathcal{P}_N\}$, with $\mathcal{P}_i \subset \mathcal{X}_{free}, i = 1, ..., N$, which maximizes a global utility function $U_I(\cdot, \cdot)$. The utility function $U_I(\mathcal{P}, \mathbf{X})$ depends on the paths $\mathcal{P}$, and on measurements already gathered by robots at positions contained in matrix $\mathbf{X} \subset \mathcal{X}_{free}$. Additionally, robots must fulfill physical and mission related-constraints. Once

robots find a solution to the finite horizon IG problem, robots follow $\mathcal{P}_i$ while taking measurements along it, and repeat the procedure again.

More formally, in this paper we propose an approximate solution to the following finite horizon problem:

$$\begin{aligned} \underset{\mathcal{P}}{\text{maximize}} \quad & U_I(\mathcal{P}, \mathbf{X}) \\ \text{subject to} \quad & \mathcal{P}_i = \{\mathbf{x}_i(t+dt), ..., \mathbf{x}_i(t+k_c dt)\} \\ & \mathbf{x}_i(t+k_t dt) = \mathbf{f}(\mathbf{x}_i(t+(k_t-1)dt), \mathbf{u}_i), \\ & ||\mathbf{x}_i(t+k_t dt) - \mathbf{x}_j(t+k_t dt)||_2^2 \geq r_s, \\ & \mathcal{G}_c(\mathcal{V}_{t+k_c dt}, \mathcal{E}_{t+k_c dt}) \text{ is connected.} \end{aligned} \tag{1}$$

with $k_t = 1, ..., k_c$, $i, j = 1, ..., N$ and $i \neq j$, and $\mathcal{P}_i \subset \mathcal{P}$.

To solve problem (1) we build on three main methods: GPs, RRT, and max-sum algorithm. Next we give an overview of the three methods.

4. Background

4.1. Gaussian Processes for Modelling Spatial Data

A GP is a collection of random variables, any finite number of which have a joint multivariate Gaussian distribution [5]. A GP is defined by $\mathbf{m}(\mathbf{x})$, the mean function, and by $k(\mathbf{x}, \mathbf{x}', \boldsymbol{\theta})$, the covariance function, over positions $\mathbf{x}$, $\mathbf{x}'$. Here we assume a zero mean prior function $\mathbf{m}(\mathbf{x})$, which implies an absence of a priori known values of the process. The covariance function depends on hyperparameters $\boldsymbol{\theta}$. We use the squared exponential (SE) [5] covariance function due to its capacity to model smooth processes. This function is determined by hyperparameters $\boldsymbol{\theta} = [\sigma_f^2, l, \sigma_n^2]^T$, being l the characteristic length-scale (informally, "how close" two positions $\mathbf{x}$ and $\mathbf{x}'$ have to be to influence each other significantly); σ_f^2 represents the maximum allowable covariance; and σ_n^2 is the variance of the noise fluctuations [5].

We use the following definitions: $\mathbf{X} = [\mathbf{x}^{[1]}, \mathbf{x}^{[2]}, \cdots, \mathbf{x}^{[n]}]^T$ is a matrix where each row corresponds to a spatial location where a robot has gathered a measurement. Vector $\mathbf{z} = [z^{[1]}, z^{[2]}, \cdots, z^{[n]}]^T$ stores the corresponding measurements. And in matrix $\mathbf{X}_* = [\mathbf{x}_*^{[1]}, \mathbf{x}_*^{[2]}, \cdots, \mathbf{x}_*^{[p]}]^T$ each row is a "probe" location – points in space where we predict the process value using the GP model. In this paper, "probe" locations correspond to points where robots aim to potentially take a measurement. Moreover, we define matrices $\mathbf{K}, \mathbf{K}_*, \mathbf{K}_{**}$ from the covariance function $k(\mathbf{x}, \mathbf{x}', \boldsymbol{\theta})$ as follows:

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}^{[1]}, \mathbf{x}^{[1]}) & \cdots & k(\mathbf{x}^{[1]}, \mathbf{x}^{[n]}) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}^{[n]}, \mathbf{x}^{[1]}) & \cdots & k(\mathbf{x}^{[n]}, \mathbf{x}^{[n]}) \end{bmatrix}, \mathbf{K}_* = \begin{bmatrix} k(\mathbf{x}^{[1]}, \mathbf{x}_*^{[1]}) & \cdots & k(\mathbf{x}^{[1]}, \mathbf{x}_*^{[p]}) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}^{[n]}, \mathbf{x}_*^{[1]}) & \cdots & k(\mathbf{x}^{[n]}, \mathbf{x}_*^{[p]}) \end{bmatrix}, \mathbf{K}_{**} = \begin{bmatrix} k(\mathbf{x}_*^{[1]}, \mathbf{x}_*^{[1]}) & \cdots & k(\mathbf{x}_*^{[1]}, \mathbf{x}_*^{[p]}) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_*^{[p]}, \mathbf{x}_*^{[1]}) & \cdots & k(\mathbf{x}_*^{[p]}, \mathbf{x}_*^{[p]}) \end{bmatrix}. \tag{2}$$

Let us emphasize that $\mathbf{K}$, $\mathbf{K}_*$, and $\mathbf{K}_{**}$ are all functions of $\boldsymbol{\theta}$ through $k(\cdot)$. We do not include this dependency to simplify notation.

From measurements $\mathbf{z}$ at positions $\mathbf{X}$, we can predict the process values $\mathbf{y}_*$ at locations $\mathbf{X}_*$ and the associated uncertainties. Vector $\mathbf{y}_*$ is a random vector with the following conditional distribution: $p(\mathbf{y}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$, where $\boldsymbol{\mu}_*$ and $\boldsymbol{\Sigma}_*$ are computed as (see [5] for more details):

$$\begin{aligned} \boldsymbol{\mu}_* &= \mathbf{m}(\mathbf{X}_*) + \mathbf{K}_*^T \mathbf{K}^{-1}(\mathbf{z} - \mathbf{m}(\mathbf{X})), \\ \boldsymbol{\Sigma}_* &= \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_*. \end{aligned} \tag{3}$$

Learning a GP model implies estimating the value of the hyperparameters $\boldsymbol{\theta}_*$ that best fit the measurements $\mathbf{z}$ at locations $\mathbf{X}$. This estimation is generally formulated as a maximum-likelihood problem, where the log-marginal likelihood (LML) with respect to $\boldsymbol{\theta}$ is maximized:

$$\boldsymbol{\theta}_* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left\{ -\frac{1}{2}\mathbf{z}^T\mathbf{K}^{-1}\mathbf{z} - \frac{1}{2}\log|\mathbf{K}| \right\}. \tag{4}$$

This is a nonlinear optimization problem that requires application of numerical optimization techniques [5].

4.2. Rapidly Exploring Random Trees

The RRT algorithm allows robots to plan paths in complex high dimensional spaces [26]. The RRT algorithm iteratively constructs a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ (tree) with a set of vertices $\mathcal{V}$ and edges $\mathcal{E}$ with the goal of finding possible trajectories starting from a state $\mathbf{x}_A$.

The algorithm is realized as follows: It draws a sample $\mathbf{x}_{rand}$ randomly from a uniform distribution defined over $\mathcal{X}_{free}$. Then it finds the nearest neighbor $\mathbf{x}_{nearest}$ (in terms of the cost-to-reach) of $\mathbf{x}_{rand}$ in the set of vertices $\mathcal{V}$. Next it simulates driving the robot from $\mathbf{x}_{nearest}$ to $\mathbf{x}_{rand}$ according to the robot's controller. In particular, it drives the robot a maximum distance η, which is a user-selected parameter that sets the maximum branch size. This results in a new state $\mathbf{x}_{new}$. If trajectory $\mathcal{E}(\mathbf{x}_{nearest}, \mathbf{x}_{new})$ does not collide with any obstacles, it adds vertex $\mathbf{x}_{new}$ and edge $\mathcal{E}(\mathbf{x}_{nearest}, \mathbf{x}_{new})$ to tree $\mathcal{G}$. This process is repeated during N_p iterations.

4.3. Max-Sum Algorithm

Let us consider a team of N robots, where each robot i can control a decision variable $\mathcal{D}_i$ that can take values from domain $\mathcal{C}_i = \{\mathcal{C}_i^{[1]}, \mathcal{C}_i^{[2]}, ..., \mathcal{C}_i^{[k_i]}\}$. In this paper, $\mathcal{C}_i^{[i']} \subset \mathcal{C}_i$ with $i' = 1, ..., k_i$ consists of a set of potential measurement locations that robot i could visit. We denote the set of variables for which we aim to solve the assignment problem as $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_N\}$. For example, for a team of three robots with identical domain size $k_i = 4$ with $i = 1, 2, 3$, a possible assignment for the variables could be $\mathcal{D} = \{\mathcal{D}_1 : \mathcal{C}_1^{[1]};\ \mathcal{D}_2 : \mathcal{C}_2^{[4]};\ \mathcal{D}_3 : \mathcal{C}_3^{[1]}\}$.

In max-sum [9], the goal of the robots is to maximize a global utility function $U(\mathcal{D}) = \sum_{i=1}^{N} U_i(\bar{\mathcal{D}}_i)$, where $U_i(\bar{\mathcal{D}}_i)$ denotes utility function of robot i, and $\bar{\mathcal{D}}_i \subset \mathcal{D}$. For instance: coming back to the previous three robots example, $\bar{\mathcal{D}}_1 = \{\mathcal{D}_1, \mathcal{D}_2\}$ implies that the utility of robot 1 depends on its own decision, and on the decision of robot 2, but not on the one from robot 3. Within this setting, we wish to find the optimal assignment $\mathcal{D}^*$ such that $U(\mathcal{D})$ is maximised: $\mathcal{D}^* = \underset{\mathcal{D}}{\operatorname{argmax}} \sum_{i=1}^{N} U_i(\bar{\mathcal{D}}_i)$.

Max-sum formulates this assignment problem as a *factor graph* [27]. A factor graph is a bi-partite graph with two types of nodes: variables and factors. Edges in this graph represent the dependencies of factors on variables. For instance, the factor graph in Figure 1 represents $U(\mathcal{D}) = U_1(\bar{\mathcal{D}}_1) + U_2(\bar{\mathcal{D}}_2) + U_3(\bar{\mathcal{D}}_3)$, where $\bar{\mathcal{D}}_1 = \{\mathcal{D}_1, \mathcal{D}_2\}$, $\bar{\mathcal{D}}_2 = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$ and $\bar{\mathcal{D}}_3 = \{\mathcal{D}_2, \mathcal{D}_3\}$.

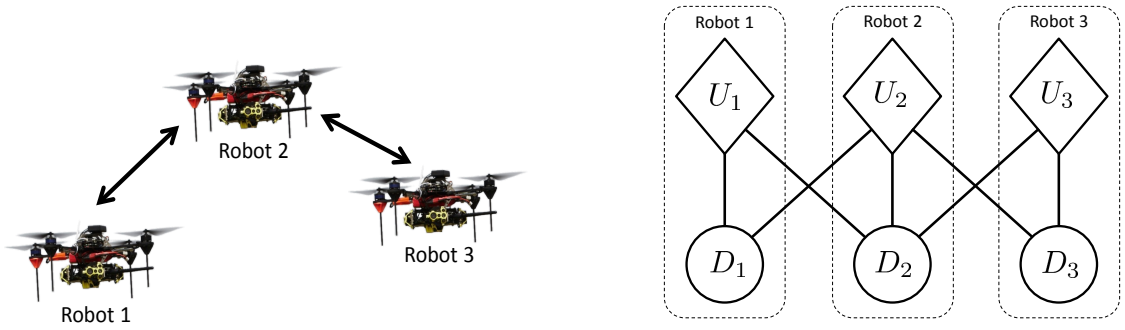
Max-sum is a message passing algorithm on factor graphs. Messages are passed along the edges of the factor graph in order to determine the variable values that maximise $U(\cdot)$. We distinguish between two types of messages:

- Factor to variable message; denoted $s_{i\to j}$. It is the maximum value of factor U_i for each possible value of $\mathcal{D}_j$.
- Variable to factor message; denoted $q_{j\to i}$. It is the maximum value of U_i neighboring factors for each possible value of $\mathcal{D}_j$.

For more details on the definition of messages we refer the reader to the original paper [9]. Provided the definitions of messages we can now summarize the execution of max-sum algorithm. First, each of the robots arbitrarily initializes $q_{j\to i}$ and sends it to its adjacent function nodes. This triggers an exchange of messages between variable and utility function nodes. The messages exchange

will continue until message values converge, or after an user-defined number of iterations. Next, each of the robots evaluates the marginal function of variable $\mathcal{D}_i$: $z_i(\mathcal{D}_i)$ [9]. Then, by simply finding $\text{argmax}_{\mathcal{D}_i} z_i(\mathcal{D}_i)$, each individual robot i is able to determine which $\mathcal{C}_i^{[i']}$, for $i' = 1, 2, .., k_i$, it should visit such that $U(\cdot)$ is maximized.

Max-sum algorithm delivers an exact solution in cases where the factor graph is acyclic, i.e. it has no loops. Otherwise, if the factor graph is cyclic, i.e., it has loops (as in, e.g., Figure 1), max-sum has been empirically shown to converge to an approximation of the exact solution [9].Moreover max-sum is robust against communication delays. Since max-sum messages are transmitted asysnchronously and do not follow a pre-defined order, max-sum is resilient to delays in the data transfer.



(**a**) Multi-robot network. (**b**) Factor graph.

Figure 1. Left: a multi-robot network, where arrows depict inter-robot communication. Right: a factor graph representing utility function $U(\mathcal{D}) = U_1(\mathcal{D}_1, \mathcal{D}_2) + U_2(\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3) + U_3(\mathcal{D}_2, \mathcal{D}_3)$. Diamonds correspond to factors, and circles correspond to variables. Note that dependencies between factors and variables are determined by the multi-robot communication network (Figure 1a).

5. Information Metric

Information metrics are used by robots to guide their movement by selecting positions $\mathbf{X}_*$ that maximize a particular information metric. Here we compare information metrics in the context of GPs based on two properties that are useful for IG. These two properties are:

- *monotonicity* as we increase the number p of potential measurement locations $\mathbf{X}_*$. That is, we are interested in information metrics that yield a higher value as we consider longer paths, i.e., paths with a higher p; and
- *submodularity* respect to p. In short, a submodular information metric offers diminishing returns as we increase p. This justifies the use of finite horizon approaches (as we do in this paper), as the amount of information obtained by increasing p becomes irrelevant from a certain value of p. For a detailed overview of submodularity applications in the context of GPs, we refer the reader to [28].

We analyze three information metrics: (i) Differential Entropy, (ii) Mutual Information Non-Measured, and (iii) Mutual Information All. To better support this analysis, we include in Figure 2 a graphical representation of some basic notation employed in this paper.

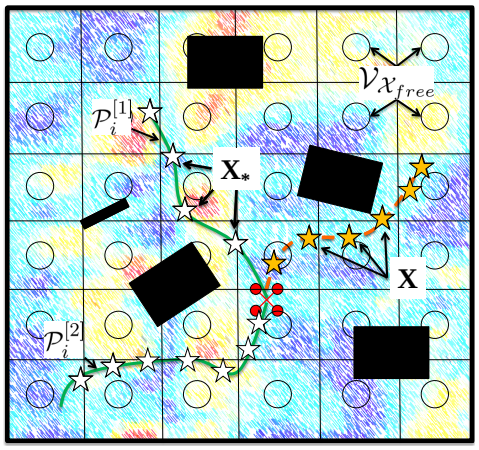


Figure 2. Graphical representation of the notation employed in this paper. We depict an scenario in which a robot i (colored red) aims to explore a process (in the background) in an environment populated with obstacles (colored black). Orange stars correspond to measurements that were previously gathered by the robot at positions $\mathbf{X}$. White stars are potential measurements locations $\mathbf{X}_*$. As in this chapter we consider a path planning mechanism, $\mathbf{X}_*$ belong to potential paths $\mathcal{P}_i^{[1]}$, $\mathcal{P}_i^{[2]}$ that could be traversed by the robot. Information metrics are utilized here to quantify the informativeness of potential paths. In addition, we also represent $\mathcal{V}_{\mathcal{X}_{free}}$, together with associated grid cells, which are needed to compute some metrics.

Differential Entropy. We denote the differential entropy of a process given by random variable $Y_{\mathbf{X}_*}$, defined at potential measurement locations $\mathbf{X}_*$, as $H(Y_{\mathbf{X}_*}|\mathbf{X})$, with $\mathbf{X}$ the location of measurements gathered by the robot up to now. $H(Y_{\mathbf{X}_*}|\mathbf{X})$ can be calculated with $H(Y_{\mathbf{X}_*}|\mathbf{X}) = \frac{1}{2}\log((2\pi e)^p|\mathbf{\Sigma}_*|)$, with $\mathbf{\Sigma}_*$ calculated with (3).

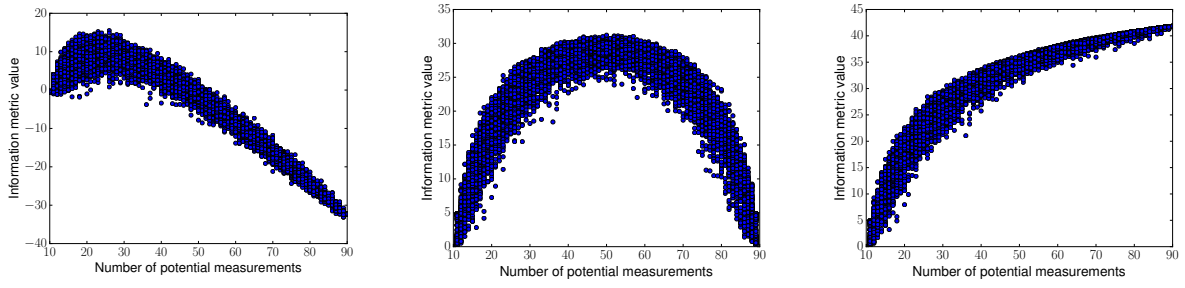
Mutual Information Non-Measured. We define `Mutual Information Non-Measured` as the MI between: a random variable $Y_{\mathbf{X}_*}$; and a random variable $Y_{\mathcal{V}_{\mathcal{X}_{free}}\setminus\{\mathbf{X}\cup\mathbf{X}_*\}}$ that represents the physical process at $\mathcal{V}_{\mathcal{X}_{free}}$ that would remain unmeasured after visiting $\mathbf{X}_*$. Note that we employ here $\mathcal{V}_{\mathcal{X}_{free}}$ instead of $\mathcal{X}_{free}$ because MI for GPs is evaluated at a set of discrete locations [28]. Therefore, to calculate MI we discretize $\mathcal{X}_{free}$ by overlaying a lattice graph with vertices $\mathcal{V}_{\mathcal{X}_{free}}$. Also note that, for $\mathbf{x} \in \mathbf{X}, \mathbf{X}_*$ and $\mathbf{x}' \in \mathcal{V}_{\mathcal{X}_{free}}$, we assume that $\mathbf{x} = \mathbf{x}'$ if $\mathbf{x}$ lies within the cell associated to $\mathbf{x}'$ (see Figure 2).

`Mutual Information Non-Measured` is given by: $I(Y_{\mathcal{V}_{\mathcal{X}_{free}}\setminus\{\mathbf{X}\cup\mathbf{X}_*\}}; Y_{\mathbf{X}_*}|\mathbf{X}) = H(Y_{\mathcal{V}_{\mathcal{X}_{free}}\setminus\{\mathbf{X}\cup\mathbf{X}_*\}}|\mathbf{X}) - H(Y_{\mathcal{V}_{\mathcal{X}_{free}}\setminus\{\mathbf{X}\cup\mathbf{X}_*\}}|\mathbf{X}, Y_{\mathbf{X}_*})$. This expression has a clear interpretation for IG: we aim to sample at locations $\mathbf{X}_*$ that yield a maximum inter-dependence with process $Y_{\mathcal{V}_{\mathcal{X}_{free}}\setminus\{\mathbf{X}\cup\mathbf{X}_*\}}$, defined at all positions in the environment that will remain unmeasured.

Mutual Information All. In this chapter we propose the use of `Mutual Information All`. This calculates the MI between a random variable $Y_{\mathbf{X}_*}$; and a random variable $Y_{\mathcal{V}_{\mathcal{X}_{free}}}$. `Mutual Information All` is given by the following expression: $I(Y_{\mathcal{V}_{\mathcal{X}_{free}}}; Y_{\mathbf{X}_*}|\mathbf{X}) = H(Y_{\mathcal{V}_{\mathcal{X}_{free}}}|\mathbf{X}) - H(Y_{\mathcal{V}_{\mathcal{X}_{free}}}|\mathbf{X}, Y_{\mathbf{X}_*})$. This metric is similar to `Mutual Information Non-Measured`, but it has an interesting property that we discuss next.

The three afore-described information metrics are submodular [29]. To study the metrics' monotinicity, we carried out a simple simulation. Specifically, we considered a one-dimensional space $\mathcal{V}_{\mathcal{X}_{free}}$ that consists of 90 equally separated positions. Then we assumed that a robot already took ten measurements, drawn from a GP at positions $\mathbf{X}$ randomly selected from $\mathcal{V}_{\mathcal{X}_{free}}$. For this setup, we evaluated the afore-described information metrics as we increase p (illustrating longer planing horizons). That is, we randomly selected from $\mathcal{V}_{\mathcal{X}_{free}}$ a number of potential measurements positions, which is given by $\mathbf{X}_*$. Results from this experiment are depicted in Figure 3, where each dot corresponds to a realization of the experiment.

From Figure 3 we can draw the following conclusion: `Differential Entropy` is non-monotonic, which goes against the principle of "information never hurts". Non-monotonocity is a property that is particularly undesirable for algorithms that aim to plan over an horizon longer than one step, as it is the case in Equation (1).



(**a**) (**b**) (**c**)

Figure 3. Evaluation of several information metrics as we increase the number of potential measurements. (**a**) Differential Entropy; (**b**) Mutual Information Non-Measured; (**c**) Mutual Information All. Figure 3c corresponds to our proposed metric.

In addition to entropy, we analyzed two uses of MI: `Mutual Information Non-Measured` and `Mutual Information All`. According to Figure 3, `Mutual Information Non-Measured` is only monotonic in the first part of the curve. This implies that `Mutual Information Non-Measured` does not allow us to plan over arbitrarily long horizons, as longer paths may result in a lower value of the information metric. Note that this property goes again against the principle of "information never hurts".

In contrast, here we propose the use of `Mutual Information All` as information metric to tackle this problem. `Mutual Information All` is monotonic (see Figure 3c), which is an ideal choice for IG tasks.

6. Distributed Multi-Robot Information Gathering Algorithm

We present in this section the algorithm that we propose to obtain an approximate solution of Equation (1). Our proposed algorithm works as follows: first, each robot plans a set of potential paths $\mathcal{P}_i$ that it could follow (Section 6.1). Specifically, each robot generates a RRT, whose root is the robot's current position. Next, robots cooperate in order to select a path that maximizes $U_I(\cdot,\cdot)$ subject to physical and mission-related constraints from Equation (1). Here $U_I(\cdot,\cdot)$ corresponds to `Mutual Information All`, as indicated in Section 5. To solve this multi-robot cooperation problem we propose the use of a DCOP algorithm: max-sum [9].

Max-sum requires that each robot knows its own set of potential paths, as well as its neighbors' set of potential paths. This set of paths we term robot's domain. To this end, we include a module that allows robots to find its neighbours, and to send its domain (Section 6.2).

Once a robot receives its neighbors' domain, it executes Max-sum (Section 6.3). Max-sum allows us to solve a combinatorial optimization problem [9]. To solve the optimization problem, each robot must evaluate all combinations of potential paths from the received domains (including its own domain). As we previously mentioned, here we consider a robot's domain as the set of all paths that are contained in the generated RRT. Since RRTs could grow large, the number of total paths could increase as well. This would result in an increase of the complexity of the combinatorial optimization, which could make the optimization computationally intractable. To solve this issue, we propose a procedure in which each robot groups the RRT paths into clusters, which reduces the robots' domain size (Section 6.2).

Max-sum outputs a cluster for each individual robot that solves (1). Then, each robot selects a path within its cluster. This is realized by evaluating `Mutual Information All` (Section 6.4).

Next, robots follow the selected paths while taking measurements along them. Measurements encode the knowledge robots have about the process of interest. Therefore, they exchange the gathered measurements through the network; i.e. they perform data fusion (Section 6.5). Finally, robots update their GPs model with the new measurements in order to improve the process model (Section 6.6).

In Figure 4 we depict a block diagram of the proposed algorithm. In particular, the diagram corresponds to the modules that each single robot executes. Modules are executed in a loop, where each loop iteration solves Equation (1). Next we explain the algorithm's modules in detail.

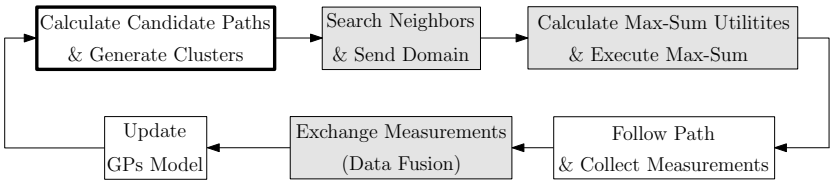


Figure 4. Algorithm block diagram. Shadowed blocks represent modules that require communication between robots.

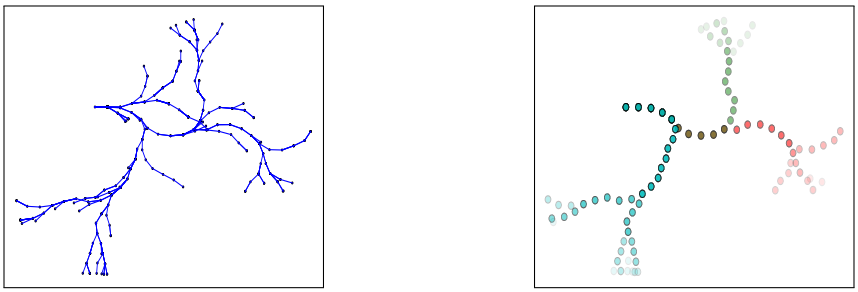
6.1. Calculate Candidate Paths and Generate Clusters

The first step of the algorithm is the computation of a set of feasible paths given $\mathbf{x}_i$ and $\mathbf{f}(\cdot)$. This is realized with the kinodynamic RRT algorithm. We introduce a constraint in RRT that guarantees collision-free paths between robots that cannot directly communicate with each other. We realize this by limiting the RRT planning horizon to a maximum distance of $(r_c - r_s)/2$, with r_c and r_s the communication and safety radius, respectively (see constraints 5 and 6 in Section 3).

Let us denote the set of paths generated by robot i with RRT as $\mathcal{P}_{i,rrt}$ (see Figure 5a). Ideally, we would like robots to exchange $\mathcal{P}_{i,rrt}$, and calculate $\mathcal{P}_i \in \mathcal{P}_{i,rrt}$ that solves (1). However, as we pointed out in Section 6 introduction, this would translate in evaluating multiple combinations of paths, which is computationally intractable. Therefore, we introduce the concept of spatio-temporal clusters. In Figure 5 we illustrate the clustering procedure with an example.

Spatio-temporal clusters give us flexibility to adapt our algorithm to the robot's computational capabilities: as we increase the number of spatial and temporal divisions, we get closer to the actual RRT. However, clusters may lead to a loss of performance when optimizing $U(\cdot)$, since robots combine several paths into a cluster during the cooperation procedure. Nevertheless, we demonstrate in Section 8.5 that performance decrease is negligible for a sufficiently large number of clusters (approx. 18 clusters for our setup).

Next we explain the clustering procedure in detail. First, we define k_t temporal horizons. Temporal horizons represent time spans of a path, which are given by the robot's motion model. For each of the temporal horizons, we extract the corresponding paths from the RRT. Then, we group paths of equal temporal horizon into k_s spatial clusters. This last step is realized running the k-means technique [30] over the complete paths.



(**a**) Tree (**b**) Clusters

Figure 5. Spatio-temporal clustering. On the left hand side we depict a Rapidly exploring Random Trees algorithm (RRT). On the right hand side we depict the clusters calculated with our proposed clustering procedure for the RRT. Specifically, we considered one temporal horizon and three spatial clusters; i.e. $k_t = 1$, $k_s = 3$, respectively. Each of the colors represent a spatio-temporal cluster.

The clustering procedure is executed by each of the robots individually, yielding $k_t \times k_s$ clusters for each robot. We denote the clusters of robot i by $C_i = \left\{C_i^{[1]}, C_i^{[2]}, ..., C_i^{[k_t k_s]}\right\}$. Note that $C_i^{[j]} \subset \mathcal{P}_{i,rrt}$ for all $j = 1, ..., k_t k_s$. Let us recall that, according to notation introduced in Section 4.3, C_i corresponds to robot i domain.

6.2. Search Neighbors and Exchange Domains

Robots move as they explore the process of interest, which results in a variation of the network topology. Therefore, we introduce a neighbors search mechanism. Robots realize this by sending an identification message with its ID. Robots that receive the identification message (see robots communication model defined in constraint 5, Section 3) add the corresponding robot's ID to its set of neighbors. Then each robot $i = 1, 2, ..., N$ sends C_i to their neighbors. The sharing of domains is the input needed to initiate the multi-robot cooperation procedure, which we explain next in detail.

6.3. Calculate Robot Utilities and Execute Max-Sum

Once robots exchange domains, they execute a max-sum algorithm to perform an assignment of clusters that approximately solves (1). Specifically, robots cooperate to find the individual cluster $\mathcal{D}_i : C_i^{[i']} \in C_i$, with $i' = 1, 2, .., k_i$, that each robot should select in order to maximize a global utility function $U(\cdot)$. Here we define $U(\cdot)$ so that it consists of two terms:

- an *information gathering* term, denoted as $\tilde{U}_I(\mathcal{D}, \mathbf{X})$, which measures the informativeness of a particular assignment of clusters $\mathcal{D}$ given previously collected measurements $\mathbf{X}$. Note that here we refer to $\tilde{U}_I(\cdot) \approx U_I(\cdot)$ as robots aim to find a joint cluster assignment, instead of a joint assignment of paths as in (1); and
- a *constraint satisfaction* term, denoted as $U_C(\mathcal{D})$, which enforces problem (1) constraints.

The combination of these terms yields our proposed utility function:

$$U(\mathcal{D}, \mathbf{X}) = \tilde{U}_I(\mathcal{D}, \mathbf{X}) - U_C(\mathcal{D}). \tag{5}$$

Let us next describe $\tilde{U}_I(\cdot)$, $U_C(\cdot)$ in more detail.

6.3.1. Information Gathering Utility—$\tilde{U}_I(\mathcal{D}, \mathbf{X})$

We define $\tilde{U}_I(\cdot)$ as the MI between $Y_{\mathcal{V}_{\mathcal{X}_{free}}}$, and a joint assignment of clusters $\mathcal{D}$, conditioned on $\mathbf{X}$. This corresponds to `Mutual Information All`, described in Section 5, and is given by the following

expression: $\tilde{U}_I(\mathcal{D}, \mathbf{X}) = I(Y_{\mathcal{V}_{\mathcal{X}_{free}}}, Y_{\mathcal{D}_1}, Y_{\mathcal{D}_2}, ..., Y_{\mathcal{D}_N}|\mathbf{X})$, with $Y_{\mathcal{D}_i}$ a GP that represents $y(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{D}_i$, $i = 1, 2, ..., N$.

Our goal is to maximize $\tilde{U}_I(\cdot)$ in a decentralized fashion. To this end, we employ max-sum algorithm, and we express $\tilde{U}_I(\cdot)$ as a sum of functions that are associated to each individual robot (see Section 4.3). By applying the chain rule for MI, and decomposing $I(Y_{\mathcal{V}_{\mathcal{X}_{free}}}, Y_{\mathcal{D}_1}, Y_{\mathcal{D}_2}, ..., Y_{\mathcal{D}_N}|\mathbf{X})$ as a difference of conditional entropies, we can express $\tilde{U}_I(\cdot)$ as:

$$\begin{aligned}\tilde{U}_I(\mathcal{D}, \mathbf{X}) &= I(Y_{\mathcal{V}_{\mathcal{X}_{free}}}, Y_{\mathcal{D}_1}, Y_{\mathcal{D}_2}, ..., Y_{\mathcal{D}_N}|\mathbf{X}) = \sum_{i=1}^{N} I(Y_{\mathcal{V}_{\mathcal{X}_{free}}}, Y_{\mathcal{D}_i}|Y_{\mathcal{D}_{i+1}}, ..., Y_{\mathcal{D}_N}, \mathbf{X}) \\ &= \sum_{i=1}^{N} H(Y_{\mathcal{V}_{\mathcal{X}_{free}}}|Y_{\mathcal{D}_{i+1}}, ..., Y_{\mathcal{D}_N}, \mathbf{X}) - H(Y_{\mathcal{V}_{\mathcal{X}_{free}}}|Y_{\mathcal{D}_i}, ..., Y_{\mathcal{D}_N}, \mathbf{X}).\end{aligned} \tag{6}$$

Equation (6) cannot be directly applied for a decentralized system that relies on local communication between robots, as robot i domain only contains information about robot i neighbors; not about all robots with a higher ID (as required in Equation (6)). We solve this issue by applying the principle of *locality* [13,24]. This allows us to assume that two random variables $Y_{\mathcal{D}_j}, Y_{\mathcal{D}_k}$ are statistically independent if spatial locations contained in $\mathcal{D}_j, \mathcal{D}_k$ are sufficiently distant. For our applications of interest, the principle of locality is a reasonable assumption as a process spatial correlation is typically much smaller than robots communication range. For example, in this paper's motivating problem of mapping a wind field, the structures (thermals) are only a few hundred meters in size. In contrast, the robots communication range tend to be in the order of kilometers.

By considering the *locality* assumption we can now formulate Equation (5) as:

$$U(\mathcal{D}, \mathbf{X}) = \sum_{i=1}^{N} H(Y_{\mathcal{V}_{\mathcal{X}_{free}}}|Y_{\mathcal{N}(\mathcal{D}_{i+1:N})}, \mathbf{X}) - H(Y_{\mathcal{V}_{\mathcal{X}_{free}}}|Y_{\mathcal{D}_i}, Y_{\mathcal{N}(\mathcal{D}_{i+1:N})}, \mathbf{X}) - U_C(\mathcal{D}_i, \mathcal{N}(\mathcal{D}_i)), \tag{7}$$

where $\mathcal{N}(\mathcal{D}_{i+1:N})$ denotes assignment variables associated just to the neighbors of robot i with a higher ID, and $\mathcal{N}(\mathcal{D}_i)$ denotes assignment variables that are associated to neighbors of the i-th robot.

6.3.2. Constraint Satisfaction Utility—$U_C(\mathcal{D})$

The role of $U_C(\cdot)$ is to satisfy that mission-related constraints (constraints 6, 7 in Section 3) are not violated. To this end, we set $U_C(\cdot) = 0$ if robots are in a configuration that is far from violating the constraints. Otherwise, we set $U_C(\cdot)$ to a value that increases within a "escape" distance r_e as robots get closer to a configuration where constraints could be violated. In case a constraint is violated we set $U_C(\cdot) = \infty$ (see Figure 6).

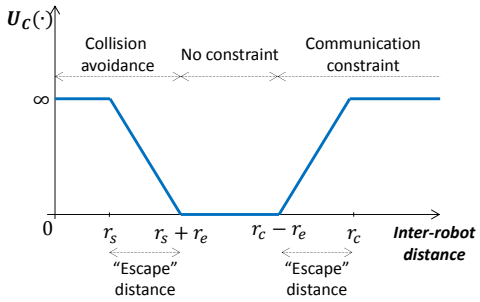


Figure 6. Graphical illustration of $U_C(\cdot)$.

Next we describe how we account for problem specific constraints:

1. *Inter-robot collision avoidance*: we penalize robots that are separated a distance smaller than $r_s + r_e$.

2. *Periodic network connectivity*: we penalize robots configurations that could lead to a disconnected network at $t + k_c\Delta t$; i.e., at the end of robots' paths.

As robots rely on local communication, it is not trivial to guarantee a periodic network connectivity [31]. Therefore, as in [23], we guarantee connectivity by forcing robots to form a minimal topology – chain topology. That is, we encourage robots to be at least connected to their peers that have an immediate lower and higher ID. This way, we can solve the communication constraint only with local communication. Note that in our proposed approach more complex mechanisms like e.g., [31,32] could be introduced in the cooperation procedure to guarantee network connectivity.

6.3.3. Path Selection

Equation (7) can be optimized in a decentralized fashion using max-sum algorithm (see Section 4.3). In our case, max-sum outputs for each robot i, $i = 1, 2, ..., N$, an optimal cluster $\mathcal{D}_i^* \in C_i$. $\mathcal{D}_i^*$ is a cluster that contains multiple paths. Therefore, robots must select a path $\mathcal{P}_i^*$ to follow from $\mathcal{D}_i^*$. This is done by calculating the MI between $Y_{\mathcal{V}_{\mathcal{X}_{free}}}$, and a random variable $Y_{\mathcal{P}_i}$ that represents all possible path assignments within the selected cluster, with $\mathcal{P}_i \in \mathcal{D}_i^*$. We condition MI on the knowledge about the selection of clusters $\left\{\mathcal{D}_j^*\right\}_{j\in\mathcal{N}_i}$ of neighboring robots $\mathcal{N}_i$, and previously gathered measurements $\mathbf{X}$. More formally, each robot i aims to find $\mathcal{P}_i^*$ such that $\mathcal{P}_i^* = \underset{\mathcal{P}_i}{\operatorname{argmax}}\, I(Y_{\mathcal{V}_{\mathcal{X}_{free}}}, Y_{\mathcal{P}_i} | \left\{\mathcal{D}_j^*\right\}_{j\in\mathcal{N}_i}, \mathbf{X})$. Let us remark that this procedure is done by each of the robots independently.

6.4. Follow Path and Collect Measurements

The output of the cooperation stage is a path $\mathcal{P}_i^*$. Then robots follow $\mathcal{P}_i^*$, and collect measurements along it. Robots add measurements values to $\mathbf{z}$, and measurements positions to $\mathbf{X}$.

6.5. Exchange Measurements (Data Fusion)

Data fusion allows robots to have a common understanding about the process of interest. In this paper we focus on multi-robot coordination strategies, and consider decentralized data fusion out of the scope of this work. Therefore, we implement a simple flooding algorithm to carry out the data fusion. Note that decentralized data fusion approaches like, e.g., [33] could also be considered. Our data fusion algorithm works works as follows: First, robots broadcast $\mathbf{z}, \mathbf{X}$ to their neighbors. Second, once a robot receives $\mathbf{z}, \mathbf{X}$ it will broadcast those again if this is the first time that they were received. This will continue till all robots receive measurements of the complete team.

6.6. Update GPs Model

Finally, robots update the GPs model with new measurements. This is done by each of the robots individually by optimizing Equation (4).

7. Computational Complexity

In this section, we carry out a study of the computational complexity of the proposed algorithm. We divide this study in three variants of the algorithm in order to highlight different aspects of the approach:

- `NoCluster`. This corresponds to the algorithm described in Section 6, but without considering the clustering method. That is, we consider there are as many clusters as paths resulting from the RRT for all time horizons (that is, as nodes in the RRT), where each cluster has a single path. This allows us to highlight the complexity in terms of the number of collected measurements, and total number of robots.
- `Cluster`. This is the algorithm described in Section 6. Here we highlight complexity reduction that results by introducing a clustering method.

- `ClusterSimplified`. This corresponds to algorithm described in Section 6 plus additional techniques that we introduce to reduce the computational complexity. These techniques: are kd-trees, sparse GPs [34], and the principle of *locality* [35]. Note that sparse GPs, and principle of *locality* are approximations that do not yield exact solutions. Nevertheless, these techniques have been shown to work well in practice in a large domain of problems as discussed in [34,35].

Next we analyze the worst-case computational complexity for the three aforementioned algorithm variants.

7.1. NoCluster

The `NoCluster` variant has three main components that define the algorithm's computational complexity: (i) RRT planner, (ii) calculation of max-sum utilities, and (iii) update of the GPs model.

- The complexity of RRT is given by $\mathcal{O}(N_p \log N_p)$, with N_p the number of RRT planner iterations [26].
- The complexity of max-sum algorithm is determined by the calculation of $H(Y_{\mathcal{V}_{\mathcal{X}_{free}}} | Y_{\mathcal{D}_i}, Y_{\mathcal{N}(\mathcal{D}_{i+1:N})}, \mathbf{X})$ in (7). This is given by the GPs regression, which is cubic on the total number of elements m [5] contained in $\mathcal{V}_{\mathcal{X}_{free}}$, $\mathcal{D}_i$, $\mathcal{N}(\mathcal{D}_{i+1:N})$ and $\mathbf{X}$. Since a robot calculates the utility of each combination of clusters (in this case, smaller or equal than N_p), the overall complexity of max-sum is $\mathcal{O}(m^3 N_p^{|\mathcal{N}_C|})$, with $|\mathcal{N}_C|$ the number of elements of $\mathcal{N}_C \triangleq \mathcal{N}(\mathcal{D}_{i+1:N})$.
- The complexity of the GPs model update in (4) is given by $\mathcal{O}(n^3 i_G)$, where n is the total number of gathered measurements, and i_G is a user-defined parameter that sets the number of iterations we allow the optimizer to calculate the GPs hyperparameters.

The complexity of the `NoCluster` variant is thus determined by max-sum, as $m >> n$, and typically $N_p >> i_G$. The benefit of using a distributed approach such as max-sum is illustrated by noticing that the complexity scales with the number of neighbors, and not with N. However, it is clearly influenced by N_p, which is typically large for robots with complex dynamics, or environments with multiple obstacles. Therefore, in order to reduce the algorithm's computational complexity we propose in this paper a concept of clustering.

7.2. Cluster

In the `Cluster` variant, the RRT structure is exploited to group N_p nodes in $k_s \times k_t$ clusters. This yields a max-sum complexity of $\mathcal{O}(m^3 (k_s k_t)^{|\mathcal{N}_C|})$. The complexity is thus now dependent on the total number of clusters, which is typically much smaller than N_p due to the tree structure. Of course, the clustering method adds additional complexity to the algorithm. However, this is negligible [30] compared to max-sum complexity. In particular, [30] has a running time of $\mathcal{O}(N_p k_s k_t d_c i_c)$, with d_c the maximum number of dimensions of a k-means state, and i_c the number of iterations of Lloyd's algorithm [30].

The complexity reduction of the `Cluster` variant is vital for an online algorithm. However, it could not be sufficient for an exploration algorithm that must run in real time. Specifically, the `Cluster` variant faces two main problems due to the complexity increase: (i) in max-sum algorithm as m grows, and (ii) in the GPs model update step as n grows.

7.3. ClusterSimplified

In order to alleviate the computational complexity of the two aforementioned problems we propose a solution that we term `ClusterSimplified`. On the one hand, we exploit the principle of *locality* to reduce the complexity, assuming that $\mathbf{x}, \mathbf{x}'$ that are far apart are uncorrelated, and therefore do not need to be considered to carry out regression. In particular, in this work we assume that $\mathbf{x}, \mathbf{x}'$ are far if $k(\mathbf{x}, \mathbf{x}', \boldsymbol{\theta}) < \sigma_n / 10$. Let us point out that this is a reasonable assumption as in this paper we

consider sensors with a negligible noise level. To efficiently search for locations that are correlated, we structure the data in a kd-tree.

The complexity of GPs regression is further alleviated by employing sparse GPs [34]. Specifically, we use the FITC method, with inducing points selected randomly from the set of potential measurements. Since the number of inducing points is typically set to be much smaller than the number of potential measurements, sparse GPs incur into an enormous reduction of complexity [34].

7.4. Summary

To finalize, we summarize in Table 1 the complexity of the three algorithm variants that we proposed in this section. Let us point out that m_s, n_s in Table 1 are the number of potential measurements, and actual measurements, respectively, which result after applying sparse GPs and *locality* approximations. Moreover, we analyzed in Table 2 the computation time of one algorithm run for a set of fix parameters that is representative of the simulations we carried out in the paper. In addition, we varied a set of parameters to account for several degrees of complexity reduction.

Motivated by a lower computational complexity and an equivalent performance, compared to other alternatives, we decided to employ our proposed variant `ClusterSimplified` in our simulations and experiments.

Table 1. Evaluation of algorithm's complexity. For clarification, let us add that typically $m_s << m$, $k_s k_t << N_p$, and $n_s << n$.

	`NoCluster`	`Cluster`	`ClusterSimplified`
Path planner	$\mathcal{O}(N_p \log N_p)$	$\mathcal{O}(N_p \log N_p)$	$\mathcal{O}(N_p \log N_p)$
Clustering method	-	$\mathcal{O}(N_p k_s k_t d_c i_c)$	$\mathcal{O}(N_p k_s k_t d_c i_c)$
Max-sum	$\mathcal{O}(m^3 N_p^{\lvert\mathcal{N}_C\rvert})$	$\mathcal{O}(m^3 (k_s k_t)^{\lvert\mathcal{N}_C\rvert})$	$\mathcal{O}(m_s^3 (k_s k_t)^{\lvert\mathcal{N}_C\rvert})$
Updating GPs	$\mathcal{O}(n^3 i_G)$	$\mathcal{O}(n^3 i_G)$	$\mathcal{O}(n_s^3 i_G)$

Table 2. Computational time required to execute one algorithm run. For the calculations, we used a set of fix parameters that is representative of the simulations we carried out in the paper. These parameters are as follows: $N_p = 1000$, $d_c = 20$, $i_c = 10$, $i_G = 10$, $m = 2500$, $|\mathcal{N}_C| = 7$. In addition, we varied parameters m_s, $k_s k_t$, n_s to account for several degrees of complexity reduction. In particular, we considered: $[m_s, k_s k_t, n_s] = [m, N_p, n]/10$, $[m, N_p, n]/50$, $[m, N_p, n]/100$, $[m, N_p, n]/200$. The set of parameters used for our simulations corresponds to the `ClusterSimplified` with a reduction factor of 100. Execution time for this set of parameters is approximately 6 s.

	`NoCluster`	`Cluster`			`ClusterSimplified`			
		/10	**/50**	**/100**	**/10**	**/50**	**/100**	**/200**
Computational time [s]	3×10^{20}	3×10^{13}	4×10^{8}	3×10^{6}	3×10^{10}	3075	6	3

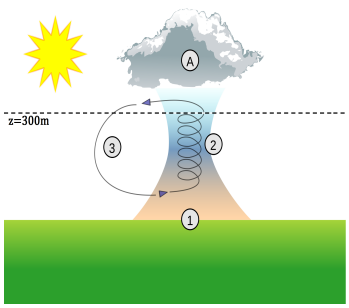
8. Simulations and Discussion of Results

8.1. Simulations Setup

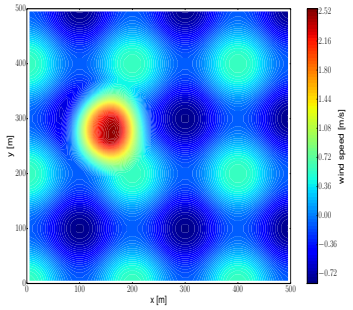
8.1.1. Generation of the Process for Exploration

We validate our algorithm in simulations in an exploration task that consists of mapping the vertical component of a wind field (see Figure 7a (By Dake (Self-made illustration) [CC BY 2.5 (http://creativecommons.org/licenses/by/2.5)], via Wikimedia Commons.)) with multiple robots. The wind field is simulated using the model proposed in [36]. The model used in [36] is an statistical model that employs data gathered from balloon and surface measurements to characterize thermals.

Furthermore we added a sinusoidal component in both x and y directions to increase the complexity of the IG task. Similar modifications were done in [3]. Figure 7b depicts the resulting wind field. This corresponds to a 500 $\times$ 500 m^2 two dimensional slice at 300 m of a three dimensional wind field. We would like to remark that the validation of our algorithm in a 2D environment (instead of in a 3D one) is motivated by two main reasons: (i) to reduce the computational complexity, and to subsequently reduce the running time of the validation in simulations, and (ii) to ease the visualization and interpretability of simulations results. The algorithm proposed in this paper is independent of the dimensionality of the environment. Therefore a 2D environment allows us to properly assess, without loss of generality, the capabilities of our algorithm to carry out an IG task.



(**a**) Thermal.



(**b**) Wind field.

Figure 7. Illustration of a thermal (**a**) and the two dimensional wind field to be explored (**b**).

8.1.2. Robot Model

We employ a simplified aircraft model that is based on modelling discussed in [37]. We made further simplifications to adapt it to a two-dimensional environment, and assumed that the wind field does not affect the aircraft's motion. These simplifications are still far from a realistic model. However, they allow us to demonstrate the effectiveness of the proposed IG approach.

Given these assumptions, the aircraft model is defined by the following equations:

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \mathbf{v}_{in}(t)\Delta t \tag{8}$$

$$\psi(t+\Delta t) = \psi(t) + \dot{\psi}(t)\Delta t, \tag{9}$$

with $\mathbf{x}(t)$ the aircraft's position, $\mathbf{v}_{in}(t)$ the aircraft's inertial velocity, and ψ the heading angle. For airspeed V, commanded flight path angle θ, and commanded bank angle ϕ, the components of the velocity $\mathbf{v}_{in}(t) = [v_x, v_y]$ and $\dot{\psi}(t)$ are given by: $v_x = V\cos\theta\cos\psi$; $v_y = V\cos\theta\sin\psi$; $\dot{\psi} = \frac{g}{V}\tan(\phi)$. Let us point out that the aircraft is fully controlled by the commanded bank angle ϕ, and flight path angle θ. For the simulations we assumed an aircraft defined by the following parameters: $\Delta t = 0.5\ s$, $V = 15\ ms^{-1}$, $g = 9.8\ ms^{-2}$, $\theta = 0$ (constant height), $\phi \in [-\pi/5, \pi/5]\ rad$.

8.1.3. Algorithm Parameters

We consider a fleet of eight aircrafts to explore the wind field. We define a communication range $r_c = 200$ m, a safety distance $r_s = 10$ m, and an escape distance $r_e = 20$ m. For the simulations we run RRT for $N_p = 1000$ iterations, and max-sum algorithm for 5 s. For the clustering algorithm, we consider four temporal horizons at $2, 5, 7, 10\ s$, and three spatial divisions. This makes 12 clusters in total for each robot.

We run Monte Carlo simulations to test our approach with a number of robots that ranges between one and eight. In particular, we considered a maximum of 4 robots for the analysis in Sections 8.2 and 8.3, and a maximum of 8 robots for the analysis in Section 8.4. The robots' initial positions is randomly set, under the requirement that the robots network is connected. For each of the algorithms

we average over 100 simulations runs. The algorithm is implemented in Python. Additionally, we use robot operating system (ROS) [38] to simulate the algorithm in a decentralized fashion.

8.2. Analysis of the Exploration Strategy

First we evaluate the performance of our proposed algorithm for an IG task that is not subject to constraints from Equation (1). This implies that robots run our algorithm with $U_C(\cdot) = 0$. This algorithm variant we term it "SBMRE Alg. No Constraints". With this study we proof the following two hypothesis:

1. The proposed cooperation procedure, which builds on MI as information metric and max-sum as decentralized coordination technique, outperforms a benchmark algorithm.
2. Our proposed algorithm scales as the number of robots in the system increases. That is, as the number of robots increases the performance gap between a benchmark and our algorithm grows.

To the best of our knowledge there are no algorithms in the literature that solve Equation (1); even in unconstrained form. Here we selected random walk as benchmark. A random walk have been shown to offer "surprisingly" good results for IG tasks [1,39]. Note that in this paper a random walk does not refer to the classic definition of a greedy random walk. Instead, here it refers to the generation of random trajectories. That is, a random walk implies that robots move independently following a random path, constrained by the robot motion, generated with RRT. The random walk neither aims to meet constraints nor to exchange measurements with the rest of the team. Let us remark that the random walk does not perform any data fusion, which implies that each of the robots only has measurements taken by itself. So, in order to obtain a fair comparison with our algorithm, which fuses data online, we perform a data fusion during post processing for the random walk benchmark.

Here we study our exploration strategy by evaluating the reduction of the root mean squared error (RMSE) after a 300 s exploration run. That is $RMSE\ Reduction[\%] = 100\frac{RMSE(t=0)-RMSE(t=300)}{RMSE(t=0)}$. We compute the RMSE with respect to a set of n_G points $\mathcal{V}_{\mathcal{X}_{free}} \in \mathcal{X}_{free}$ that correspond to nodes of an overlaid lattice graph with a spatial resolution of 10 m. We use these n_G points to compare the difference between our estimate $\boldsymbol{\mu}_*$, which is the result of GPs regression given $\mathbf{z}, \mathbf{X}$, and ground truth $\mathbf{y}_G(\mathbf{X}_G)$, with $\mathbf{X}_G := \mathcal{V}_{\mathcal{X}_{free}}$. This yields the following expression for the RMSE:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n_G} \left(\boldsymbol{\mu}_*^{[i]} - \mathbf{y}_G^{[i]}\right)^2}{n_G}}. \tag{10}$$

We depict in Figure 8a the RMSE reduction for one, two, three and four robots. First fact that we observe is that our algorithm offers an increase of performance with respect to a random walk of a 6% with one robot, and increases up to a 20% with four robots. Next fact is that the gap between our algorithm's performance and a random walk increases as we add more robots to the team. According to results from Figure 8a we can confirm our two hypothesis.

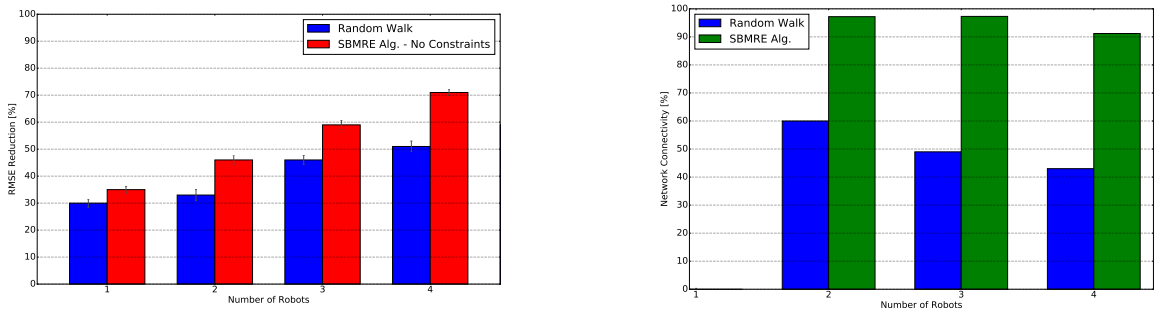


Figure 8. Root Mean Squared Error (RMSE) reduction and network connectivity. (**a**) RMSE reduction during an exploration task as we increase the number of robots in the system. (**b**) Percentage of iterations in which the network fulfills a periodic connectivity constraint.

8.3. Analysis of the Multi-Robot Coordination Strategy

We demonstrated our algorithm's cooperation capabilities to gather information. Next we analyze our algorithm's coordination capabilities to meet problem specific constraints from Equation (1). Therefore, here we proof two hypothesis, which correspond to the inter-robot constraints considered in this work. These are the following:

1. Our algorithm meets the collision avoidance constraint, and outputs collision-free trajectories.
2. The network connectivity constraint is fulfilled, and our algorithm guarantees a higher connectivity than a random walk benchmark.

8.3.1. Collision Avoidance

This section evaluates the collision avoidance capabilities of our algorithm. In particular, we calculate the percentage of time that the constraint is not met during all simulation runs, which we obtained by evaluating the distance between each pair of robots for each iteration. In Figure 9 we depict one example of the inter-robot distances during one algorithm execution. Figure 9 helps us to understand how robots coordinate to avoid collisions. Moreover it illustrates a potential collision between robots around iteration number 210 and 220, as the inter-robot distance is smaller than the safety distance $r_s = 10$ m.

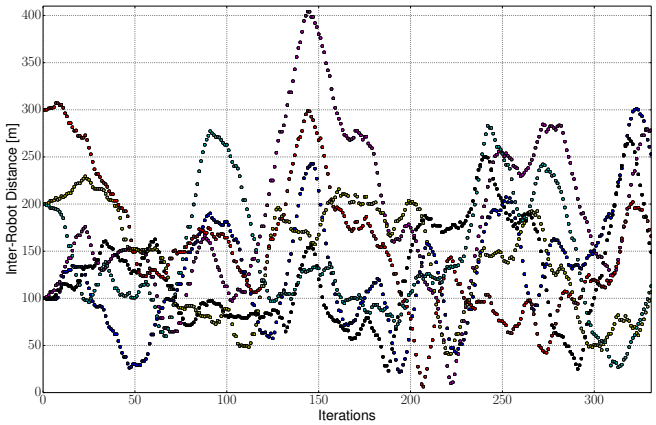


Figure 9. Inter-robot distance during one illustrative execution of our algorithm. Each color represents the distance between a different pair (6 different pairs) of robots for a 4-robot team.

For all simulation runs, the percentage of time that the collision avoidance constraint is not met is 0.16%. Let us remark here that a low percentage is still possible as the escape distance could be violated. In this sense, local safety measures and obstacle avoidance mechanisms [40] could be employed to solve such conflicts. Specifically, in [40] the authors present a collision avoidance algorithm for multiple aerial vehicle systems than functions in real-time. The proposed algorithm is based on the 3D-Optimal Reciprocal Collision Avoidance (ORCA) algorithm, and considers dynamic constraints of the UAV model and static obstacles.

A fundamental feature of our algorithm is that the violation of the collision avoidance constraint can be detected in advance by evaluating the robot individual utility function. In case of a potential collision, an algorithm like the one proposed in [40] could be executed. In contrast, a random walk has no means to anticipate a future possible collision without an external collision avoidance system.

8.3.2. Network Connectivity

Next we evaluate the fulfillment of the network connectivity constraint. To this end, we calculate the percentage of iterations in which the network is not connected at the end of robots' paths (during max-sum execution) for all simulation runs. This means that there are robots or subsets of robots that cannot communicate with the rest of the team, and therefore they violate the periodic connectivity constraint (constraint 7 in Section 3). As pointed out before, non-connectivity is an undesirable characteristic for most applications [23,41].

Figure 8b shows the network connectivity for our algorithm and a random walk. Our proposed algorithm achieves a network connectivity that ranges between 91% and 98%. In contrast, the random walk achieves a connectivity that ranges between 42% and 60%.

8.4. Analysis of the Algorithm's Scalability with an Increasing Number of Robots

A fundamental aspect of any multi-robot algorithm is its scalability as the number of robots increases. In this section we analyze the scalability in terms of the computational load that each robot must deal with. As stated in Section 7, the computational load is determined by the calculation of utilities in max-sum. Specifically, computational load scales exponentially with $|\mathcal{N}_C|$ – number of neighboring robots with which each individual robot must cooperate.

Therefore, we analyze $|\mathcal{N}_C|$ as we increase the number of robots in the system from 2 to 8 robots. We compare our algorithm against a system that requires full connectivity of the network, or in another words, a system that is centralized. We depict in Figure 10 simulation results.

We can conclude according to Figure 10 that, in a fully connected/centralized system, $|\mathcal{N}_C|$ increases linearly, which results in an exponential increase of the computational load per robot (see Table 1). In contrast, our distributed algorithm only presents a slight increase in $|\mathcal{N}_C|$ as we increase the number of robots. This results in an slight increase of the computational load per robot.

To summarize: we can conclude that our algorithm scales with the number of robots as the computational load per robot only increases slightly.

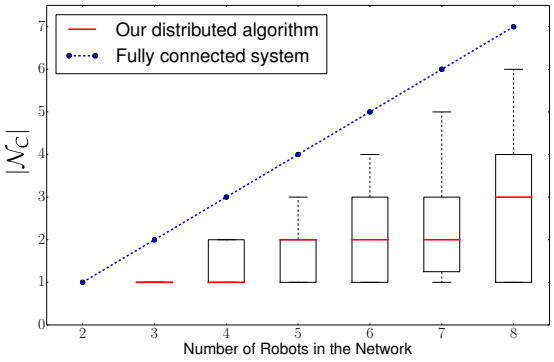


Figure 10. Algorithm's scalability for an increasing number of robots. Scalability is measured in terms of the computational load per robot, which is exponential in $|\mathcal{N}_C|$. We compare a fully connected/centralized system against our distributed solution.

8.5. Analysis of the Clustering Procedure

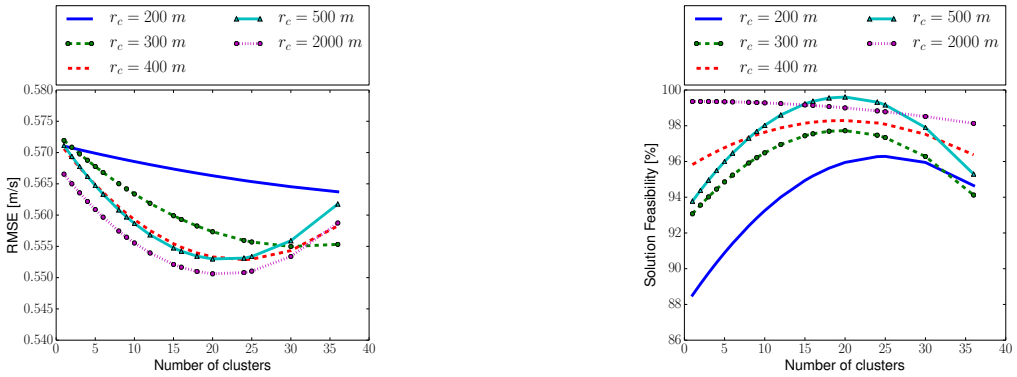
The evaluation of the exploration and coordination strategies illustrate the effectiveness of our approach to solve problem (1): performing an IG task with multiple robots while fulfilling problem specific constraints. In this section we evaluate the algorithm's sensitivity to changes in parameters values. In particular, we focus the study on the two most relevant parameters: number of spatial-temporal clusters, and communication radius. For these two parameters we analyze: (i) the resulting RMSE between estimation and ground truth after three iterations of the algorithm, and (ii) the solution feasibility; i.e., how often the algorithm is able to find a solution that meets the constraints imposed in Equation (1).

We carry out the analysis for an environment that measures 1000×1000 square meters, with a wind field that is similar to the one shown in Figure 7b but it contains two thermals. For that scenario, we run 5000 Monte Carlo simulations with randomly chosen parameters. Specifically, the number of clusters ranges from 1 to 36, and we consider a communication radius of 200, 300, 400, 500 and 2000 meters. Let us also add that we let max-sum run for 180 seconds each algorithm iteration in order to being able to calculate all utilities for up to 25 clusters.

Figure 11 depicts the results of the parameters analysis. The depicted curves are the result of a quadratic curve fitting done on the original data. From Figure 11 we can extract four main conclusions:

1. The softer the constraints, i.e., a larger communication radius, the better the algorithm's performance both in terms of RMSE and solution feasibility.
2. Our algorithm's performance increases, i.e., lower RMSE and higher solution feasibility are achieved, as we increase the number of clusters up to approximately 18 clusters. This demonstrates that the larger the number of clusters, the better we represent the original RRT, which translates into a more efficient multi-robot cooperation.
3. The performance of the algorithm remains approximately constant for a number of clusters that ranges between 18 and 25. In another words: adding new clusters does not improve the representability of the original RRT, since clusters start containing paths that are very similar. This property leads to an enormous reduction of the algorithm's computational complexity as indicated in Section 7.
4. Performance of the algorithm decreases with a number of clusters greater tham 25, for our particular setup. It is essentially due to an insufficient running time for max-sum to converge, which results in a suboptimal solution. This result emphasizes the importance of point 3, since according to Figure 11 with a number of clusters equal to approximately 18, for our setup, we obtain the best performance both in terms of RMSE and solution feasibility.

This section concludes the analysis of the algorithm in simulations. Next we present experimental results.



(**a**) RMSE. (**b**) Solution Feasibility.

Figure 11. Algorithm's performance as we vary the two most relevant parameters: number of spatial-temporal clusters, and communication radius. For these two parameters we analyze: (**a**) the resulting RMSE, calculated with (10), after three iterations of the algorithm, and (**b**) the solution feasibility; i.e. how often the algorithm finds a solution that meets constraints from Equation (1).

9. Experiments and Discussion of Results

To validate the algorithm we carried out a field experiment with flying robots. Specifically, we explored a simulated two-dimensional wind field with quadcopters emulating a fixed-wing aircraft's dynamics.

In this experiment we aim to proof the following statements:

1. Our system is able to perform active IG online, according to the measured values.
2. Our system is robust against inaccuracies in robots' position.

Next we describe in detail the experimental setup and results.

9.1. Experimental Setup

9.1.1. Wind Field model

We simulated a wind field, instead of measuring an actual field in order to simplify the overall experiment. This allows us to abstract ourselves from the particular sensor characteristics, and evaluate the algorithm's performance in a real scenario.

The wind field corresponds to a scaled down version of the one described in Section 8.1 (see Figure 7). Specifically, we reduced the size of the environment by a factor 10. This results in a wind field over an area of 50×50 square meters. Here we set $\Delta t = 0.2$, $r_c = 20$ m, $r_s = 5$ m and $r_e = 3$ m to account for a smaller environment. For the rest of parameters, we employ the same values as in Section 8.1.

9.1.2. System Architecture

To explore the afore-described wind field, we propose a system architecture that is composed of the following main elements: (i) quadcopters, (ii) a central computer, and (iii) a Real-Time Kinematic navigation for global positioning systems (GPS-RTK). Next we provide details of each of the components.

Quadcopters. We use three quadcopters that emulate the dynamics of a simple fixed-wing aircraft. In particular, we employ Equation (8) to plan the quadcopters trajectories. A trajectory can be

represented as a set of waypoints that the quadcopters can follow using their onboard controllers. This way, quadcopters will perform a flight path that is close to the one performed by a fixed-wing aircraft. It is true that this solution does not fully emulate the dynamics of a fixed-wing aircraft. Nevertheless it is a first step towards the validation of our algorithm in a field experiment.

Figure 12a shows one of the quadcopters used for the experiment. Quadcopters are a modified version of an AscTec Hummingbird from Ascending Technologies. We equipped them with a Raspberry Pi 2 Model B that sends commands to the quadcopter's onboard controller. Note that the core of the algorithm runs in a central computer due to the insufficient computational capabilities of a Raspberry Pi.

Central computer. A laptop situated outside the exploration area monitors the complete system, and runs the core of the algorithm. Specifically, it executes the algorithm that coordinates robots, and then sends waypoints to quacopters. Communication between quadcopters and the central computer is realized using Wi-Fi. Quadcopters will then fly to the commanded waypoints, using the onboard controller that runs in the Raspberry Pi. It is important to remark that the algorithm runs in a distributed fashion where each quadcopter runs in a separate software module – ROS node.

GPS-RTK. Quadcopters are also equipped with a GPS-RTK [42]. Specifically, we mounted the Piksi 1 modules from Swift Navigation. GPS-RTK allows us to achieve a sub-meter-level accuracy in the position.

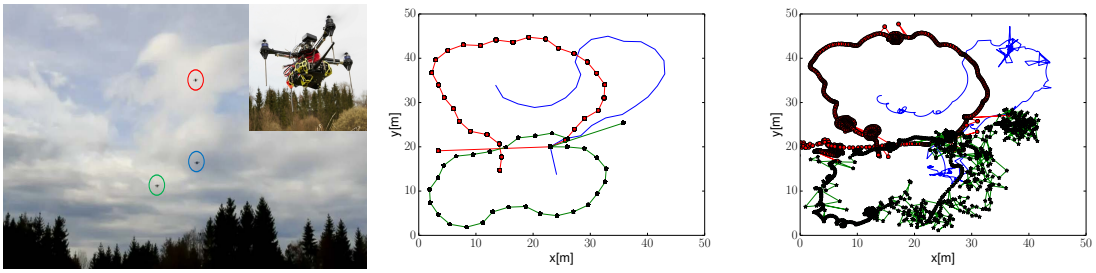
9.2. Experimental Results

Here our goal is to evaluate: (i) the robots' trajectories, (ii) the process reconstruction and the remaining uncertainty after the exploration task, and (iii) the RMSE between the process reconstruction and ground truth. For the last one we compare runs with one and three quadcopters to highlight the benefits of a multi-robot system.

9.2.1. Robots Trajectories

First, we depict in Figure 12 the trajectories that quadcopters flew during the exploration run. Figure 12b corresponds to robots' nominal position. We can observe that the shape of the trajectories resembles those of a simple fixed-wing aircraft. Moreover, robots cover the complete exploration area; except the bottom right corner. This was due to battery life constraints, which did not let robots to complete the exploration task.

Figure 12c depicts quadcopters positions as output from the GPS-RTK system. Trajectories are similar to the nominal ones. However, we observe inaccuracies in the position solution. For example, we could concentrate in Figure 12c on a large concentration of dots at coordinates $x = 15$, $y = 45$ meters. These dots correspond to a single commanded waypoint where a quadcopter tries to stay at. Ideally, we would like quadcopters to hold their position. However, this is not possible due to the combined effect of innacuracies in the robot's controller, which relies on external sensors to calculate its position, and the GPS-RTK solution. Nevertheless, let us emphasize that these inaccuracies in position do not result in an inaccurate estimation of the wind field for the chosen size of the environment, as we will show next.

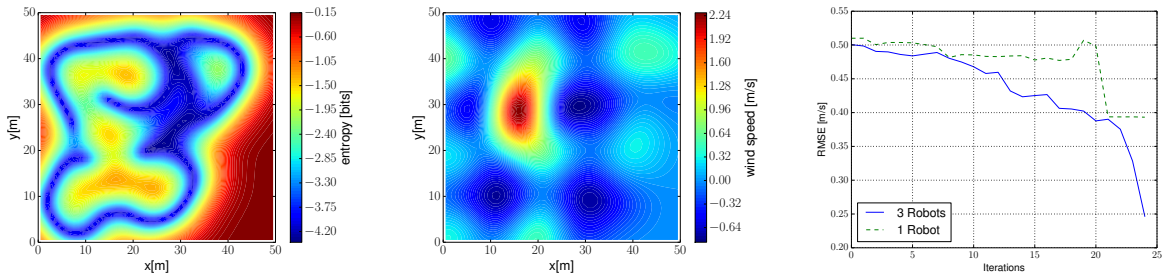


(**a**) (**b**) (**c**)

Figure 12. Nominal position and actual position of the quadcopters during the exploration task. Each of the three quadcopters is represented with a different color. (**a**) Quadcopters during the experiment. (**b**) Nominal position. (**c**) Actual position.

9.2.2. Wind Field Estimation

We illustrate in Figure 13b the estimated wind field. It corresponds to the mean prediction of the GPs at each of the positions of the environment given the collected measurements. The estimated wind field can be compared to the ground truth (depicted in Figure 7b). First, we observe that estimation and ground truth are almost identical, and we can easily identify the thermal. This exemplifies the algorithm robustness to uncertainty in robot's position. Second, we notice that the estimation is worse at those areas that were not covered during the exploration run; i.e., bottom right corner (also noticeable in Figure 13a). However, even on that area the algorithm achieves a decent reconstruction accuracy.



(**a**) Process entropy (**b**) Process reconstruction (**c**) RMSE

Figure 13. (**a**,**b**) Process entropy and reconstruction after performing an exploration run. (**c**) RMSE between estimate and ground truth for one and three quadcopters.

9.2.3. Error between Estimate and Ground Truth

We show in Figure 13c an evaluation of the RMSE between estimate and ground truth resulting from the field experiment. We show curves for one and three robots running the algorithm proposed in this work. As we showed in simulations, the system with three robots achieves a much lower RMSE compared to one robot. Specifically, three robots achieve a three-fold improvement compared to one robot. This confirms the benefits, in terms of efficiency, of a multi-robot system.

10. Conclusions and Future Work

The paper presents an approach for multi-robot information gathering. It considers GPs as underlying model of the process to explore, information utilities for active perception, and the max-sum algorithm for multi-robot cooperation. The approach extends the state of the art by accounting for the motion constraints of the robots. This is realized through the use of motion planners such as

RRT, which are able to handle such constraints. The method is able as well to handle mission team constraints such as network connectivity and collision avoidance restrictions. We achieved this by including additional terms into the utility functions in max-sum.

The whole approach is distributed, not requiring a central entity for processing. All the decision-making is decentralized, and, in our current implementation, only the data fusion component requires a broadcast mechanism at the network level (even though the system can work if the network connectivity is not fulfilled). As future work, we will consider decentralized data fusion approaches for GPs, as in [13,14], for a fully decentralized system.

The approach has been validated in simulation for the exploration of a wind field. We have also tested the methods in experiments with robots for the same application. The results show how the cooperation allows for a more efficient exploration, more evident when the number of robots grow. Furthermore, the results show how the approach can handle constraints that are relevant for real scenarios, in particular maintaining the network connectivity in the fleet.

One of the limitations of the presented application is the use of a fixed chain network topology, which constraints the ability of the fleet to explore. More dynamic and flexible network topologies would definitely allow for better information gathering efficiency. Furthermore, the connectivity model is a simplification, and more details regarding the actual physical layer and communication technology would be needed to model the communication constraints. Please notice that this is not a restriction of the decision making method itself.

The vehicle models employed in this work are a simplified version of a fixed-wing aircraft. We plan to extend those models to full 3D models that consider also aerodynamic effects, as a next step to apply the approach for the autonomous soaring of gliders. We will consider exploration in 3D, and combining the exploration techniques presented with the exploitation of the wind information for longer endurance of the flight. Exploitation terms can be easily included into our utility functions. The analysis of different combinations and weighting of the terms that compose the utility function is also a venue for future work. One additional aspect to consider will be the effect of the uncertainties in the wind field over the finally executed paths, and how to include this also into the constraints of the system.

Author Contributions: Investigation, A.V. and L.M.; Methodology, A.V., Z.X. and L.M.; Software, A.V.; Supervision, Z.X. and L.M.; Validation, A.V.; Writing—original draft, A.V. and L.M.; Writing—review & editing, A.V. and L.M.

Acknowledgments: The work of L.M. is partially funded by the Spanish Ministry of Science, Innovation and Universities Project COMCISE (RTI2018-100847-B-C22, MCIU/AEI/FEDER, UE).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Viseras, A.; Wiedemann, T.; Manss, C.; Magel, L.; Mueller, J.; Shutin, D.; Merino, L. Decentralized multi-agent exploration with online-learning of Gaussian processes. In Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 16–21 May 2016; pp. 4222–4229.
2. Hitz, G.; Galceran, E.; Garneau, M.È.; Pomerleau, F.; Siegwart, R. Adaptive continuous-space informative path planning for online environmental monitoring. *J. Field Robot.* **2017**, *34*, 1427–1449.
3. Lawrance, N.R.; Sukkarieh, S. Autonomous exploration of a wind field with a gliding aircraft. *J. Guid. Control Dyn.* **2011**, *34*, 719–733.
4. Stranders, R.; Fave, F.M.D.; Rogers, A.; Jennings, N.R. A decentralised coordination algorithm for mobile sensors. In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, Atlanta, GA, USA, 11–15 July 2010; AAAI Press: Palo Alto, CA, USA, 2010; pp. 874–880.
5. Rasmussen, C.E.; Williams, C.K. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*; The MIT Press: Cambridge, MA, USA, 2005.
6. Singh, A.; Krause, A.; Guestrin, C.; Kaiser, W.J. Efficient Informative Sensing using Multiple Robots. *J. Artif. Intell. Res.* **2009**, *34*, 707–755.

7. Choi, H.L.; Lee, S.J. A Potential-Game Approach for Information-Maximizing Cooperative Planning of Sensor Networks. *IEEE Trans. Control Syst. Technol.* **2015**, *23*, 2326–2335.
8. Doo-Hyun, C.; Jung-Su, H.; Su-Jin, L.; Sunghyun, M.; Han-Lim, C. Informative Path Planning and Mapping with Multiple UAVs in Wind Fields. In Proceedings of the 13th International Symposium on Distributed Autonomous Robotic Systems, London, UK, 7–9 November 2016.
9. Farinelli, A.; Rogers, A.; Petcu, A.; Jennings, N.R. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems—Volume 2, Estoril, Portugal, 12–16 May 2008; pp. 639–646.
10. Oliehoek, F.A.; Whiteson, S.; Spaan, M.T. Exploiting structure in cooperative Bayesian games. *arXiv* **2012**, arXiv:1210.4886.
11. Philip, G.; Schwartz, H.M.; Givigi, S.N. Cooperative Exploration Using Potential Games, Manchester, UK, 13–16 October 2013. In Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics, Manchester, UK, 13–16 October 2013; pp. 2384–2389.
12. Viseras, A.; Xu, Z.; Merino, L. Distributed Multi-Robot Cooperation for Information Gathering under Communication Constraints. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 1267–1272.
13. Ouyang, R.; Low, K.H.; Chen, J.; Jaillet, P. Multi-robot active sensing of non-stationary Gaussian process-based environmental phenomena. In Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems, Paris, France, 5–9 May 2014; pp. 573–580.
14. Chen, J.; Low, K.H.; Yao, Y.; Jaillet, P. Gaussian Process Decentralized Data Fusion and Active Sensing for Spatiotemporal Traffic Modeling and Prediction in Mobility-on-Demand Systems. *IEEE Trans. Autom. Sci. Eng.* **2015**, *12*, 901–921, doi:10.1109/TASE.2015.2422852.
15. Viseras, A.; Shutin, D.; Merino, L. Robotic Active Information Gathering for Spatial Field Reconstruction with Rapidly-Exploring Random Trees and Online Learning of Gaussian Processes. *Sensors* **2019**, *19*, 1016.
16. Choi, H.L.; How, J.P. Continuous trajectory planning of mobile sensors for informative forecasting. *Automatica* **2010**, *46*, 1266–1275, doi:10.1016/j.automatica.2010.05.004.
17. Levine, D.; Luders, B.; How, J.P. Information-rich Path Planning with General Constraints using Rapidly-exploring Random Trees. In Proceedings of the AIAA Infotech@Aerospace Conference, Garden Grove, CA, USA, 19–21 June 2012.
18. Hollinger, G.A.; Sukhatme, G.S. Sampling-based robotic information gathering algorithms. *Int. J. Robot. Res.* **2014**, *33*, 1271–1287.
19. Chung, J.J.; Lawrance, N.R.J.; Sukkarieh, S. Learning to soar: Resource-constrained exploration in reinforcement learning. *Int. J. Robot. Res.* **2014**, *34*, 158–172.
20. Nguyen, J.L.; Lawrance, N.R.J.; Fitch, R.; Sukkarieh, S. Real-time path planning for long-term information gathering with an aerial glider. *Auton. Robots* **2016**, *40*, 1017–1039, doi:10.1007/s10514-015-9515-3.
21. Schlotfeldt, B.; Thakur, D.; Atanasov, N.; Kumar, V.; Pappas, G.J. Anytime Planning for Decentralized Multirobot Active Information Gathering. *IEEE Robotics Autom. Lett.* **2018**, *3*, 1025–1032, doi:10.1109/LRA.2018.2794608.
22. Kai-Chieh, M.; Zhibei, M.; Lantao, L.; Sukhatme, G.S. Multi-Robot Informative and Adaptive Planning for Persistent Environmental Monitoring. In Proceedings of the 13th International Symposium on Distributed Autonomous Robotic Systems, London, UK, 7–9 November 2016.
23. Gan, S.K.; Fitch, R.; Sukkarieh, S. Online decentralized information gathering with spatial–temporal constraints. *Auton. Robots* **2014**, *37*, 1–25.
24. Stranders, R.; Farinelli, A.; Rogers, A.; Jennings, N.R. Decentralised coordination of mobile sensors using the max-sum algorithm. In Proceedings of the 21st International Joint Conference on Artifical Intelligence, Pasadena, CA, USA, 11–17 July 2009; Morgan Kaufmann Publishers Inc.: Burlington, MA, USA, 2009; pp. 299–304.
25. Muppirisetty, L.S.; Svensson, T.; Wymeersch, H. Spatial wireless channel prediction under location uncertainty. *IEEE Trans. Wirel. Commun.* **2015**, *15*, 1031–1044.
26. LaValle, S.M.; Kuffner, J.J. Randomized kinodynamic planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400.
27. Kschischang, F.R.; Frey, B.J.; Loeliger, H.A. Factor graphs and the sum-product algorithm. *IEEE Trans. Inf. Theory* **2001**, *47*, 498–519.
28. Krause, A.; Guestrin, C. Submodularity and its applications in optimized information gathering. *ACM Trans. Intell. Syst. Technol. (TIST)* **2011**, *2*, 32.
29. Cover, T.M.; Thomas, J.A. *Elements ofInformation Theory*; John Wiley & Sons: Hoboken, NJ, USA, 2012.

30. Lloyd, S. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **1982**, *28*, 129–137.
31. Michael, N.; Zavlanos, M.M.; Kumar, V.; Pappas, G.J. Maintaining connectivity in mobile robot networks. In *Experimental Robotics*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 117–126.
32. Yang, P.; Freeman, R.A.; Gordon, G.J.; Lynch, K.M.; Srinivasa, S.S.; Sukthankar, R. Decentralized estimation and control of graph connectivity for mobile sensor networks. *Automatica* **2010**, *46*, 390–396.
33. Chen, J.; Low, K.H.; Tan, C.; Oran, A.; Jaillet, P.; Dolan, J.M.; Sukhatme, G.S. Decentralized data fusion and active sensing with mobile sensors for modeling and predicting spatiotemporal traffic phenomena. *arXiv* **2012**, arXiv:1206.6230.
34. Quiñonero-Candela, J.; Rasmussen, C.E. A unifying view of sparse approximate Gaussian process regression. *J. Mach. Learn. Res.* **2005**, *6*, 1939–1959.
35. Guestrin, C.; Krause, A.; Singh, A.P. Near-optimal sensor placements in Gaussian processes. In Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 7–11 August 2005; pp. 265–272.
36. Allen, M.J. Updraft model for development of autonomous soaring uninhabited air vehicles. In Proceedings of the Forty Fourth AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, USA, 9–12 January 2006.
37. Renzaglia, A.; Reymann, C.; Lacroix, S. Monitoring the Evolution of Clouds with UAVs. In Proceedings of the IEEE International Conference on Robotics and Automation, Stockholm, Sweden, 16–21 May 2016.
38. Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; Wheeler, R.; Ng, A. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009.
39. Viseras, A.; Shutin, D.; Merino, L. Online information gathering using sampling-based planners and GPs: An information theoretic approach. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 123–130.
40. Alejo, D.; Cobano, J.; Heredia, G.; Ollero, A. Optimal reciprocal collision avoidance with mobile and static obstacles for multi-UAV systems. In Proceedings of the 2014 International Conference on Unmanned Aircraft Systems (ICUAS), Orlando, FL, USA, 27–30 May 2014; pp. 1259–1266.
41. Hollinger, G.; Singh, S. Multi-robot coordination with periodic connectivity. In Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA), Anchorage, AK, USA, 3–7 May 2010; pp. 4457–4462.
42. Misra, P.; Enge, P. *Global Positioning System: Signals, Measurements and Performance Second Edition*; Ganga-Jamuna Press: Lincoln, MA, USA, 2006.